International Journal of Current Engineering and Technology
E-ISSN 2277 – 4106, P-ISSN 2347 – 5161

Available at http://inpressco.com/category/ijcet

*Research Article*

# New Local Sequence Alignment Algorithm with Adaptive Seeds and Maximum Match Subsequence (ASMMS)

**Suchindra Suchindra† and Preetam Nagaraj†**

†[1]Department of Engineering, National Institute of Mental Health and Neurosciences, Karnataka State Govt, Bangalore, India
‡[2]Department of Engineering, IBM, Bangalore, India

***Abstract***

*Sequence alignment is an important step in many fields today, from genome research to Pharma. It is famously used to determine how closely two sequences are related and at times to see how little they differ, example finding one's relatives. In computational biology, there are algorithms developed over time to not only align two sequences quickly but also to get biological data. The very first algorithms developed were based off a technique called Dynamic Programming, which were slow but produced optimal alignment. To improve speed, more algorithms today are based off heuristic approach, sacrificing sensitivity. In this paper, we are going to improve on a heuristic algorithm which is accepted to be published in the Journal of Biosciences and Engineering (BIOEJ). This new algorithm appropriately called ASMMS, stands for Adaptive Seeds and Maximal Match Subsequence local alignment algorithm. The algorithm is based on suffix tree data structure, but to improve sensitivity, we employ adaptive seeds, and perfect match seeds in between the already identified maximal matches. We tested this algorithm on a randomly generated sequences, and small dataset of genes where the sequence length ranged up to 500 thousand, our algorithm performed better than the rest.*

***Keywords:*** *Bioinformatics, Dynamic, Heuristic, Seed*

## 1. Introduction

In computational biology or bioinformatics, a sequence is either an RNA, DNA or a protein string made up of their representative character set. DNA ( A, C, G, T) , RNA ( A, C, G, U) and protein molecules (A, R, N, D, C, Q, E, G, H, I, L, K, M, F, P, S, T, W, Y, V) can be re represented as strings of letters from their alphabet set (Reddy, 2009) .

A sequence alignment is a way of arranging these sequences made of their representative characters with an objective to find the regions of '*similarity*'. These similarities would then provide additional information on the functional, structural, evolutionary, and other interest between the sequences in study. Aligned sequences are represented in rows, stacked up one on top of the other as shown below in Figure 1.

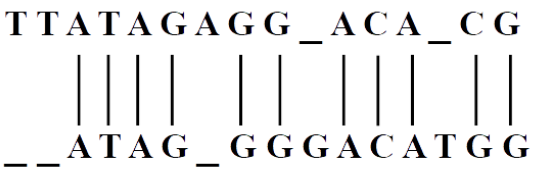


**Fig 1** Example of a sequence alignment (Reddy and Fields, 2020)

In Figure 1, there are regions, where two sequences are aligned perfectly, these regions are called 'similar region. In some regions, special characters '-', known as indels are present. These indels represents a mutation (change) or could be looked at as deletion from the other sequence's point of view (Haque, *et al*, 2009).

Pairwise sequence alignment is used to find conserved regions in two sequences. Multiple sequence alignments are used to find common regions in more than 2 sequences. Pairwise Sequence alignment (Haque, *et al*, 2009) represents at times the first step in many bioinformatics solutions. In many multiple alignment algorithms this remains the first step, especially linear multiple sequence alignment algorithms.

Pairwise sequencing as in Figure 1, is alignment between 2 sequences of the same kind, DNA, RNA or Protein. The alignment would then throw some knowledge on the divergence of one sequence over the other in some cases or similarity in some cases.

Pairwise sequence alignment can be classified into local sequence and global sequence alignment. Local sequence alignment finds the best approximate subsequence match within two given sequences while the global sequence alignment takes the entire sequence into consideration (Reddy and Fields, 2020).

Local sequence alignments are therefore designed to search subregions within the two sequences. For finding similar (biologically conserved) regions, which may or

*Corresponding author's ORCID ID: 0000-0000-0000-0000
DOI: https://doi.org/10.14741/ijcet/v.13.2.12

may not be preserved in order or orientation, local sequence alignment is very useful. It is typically used to find similarity between two divergent sequences and for fast database searches for similar sequences (Reddy and Fields, 2020). Since, it is trying to find subregions and not the sequence in its entirety, local sequence alignment usually takes less computation time when compared to global sequence alignment algorithms (Reddy and Fields, 2020).

Some of the most popular local sequence algorithms are Smith-Waterman (Smith and Waterman, 1981), FASTA (Lipman and Peason, 1985), BLAST (Altschul, *et al*, 1990), GappedBLAST Altschul, *et al*, 1997), BLASTZ (Schwartz, *et al*, 2000), PatternHunter (Ma, *et al*, 2002), YASS (No and Kucherov, 2005), LAMBDA (Singer, *et al*, 2014), USearch (Edgar, 2010), LAST (Kiełbasa, 2011), and ALLAlign (Wachtel, 2016).

Popular global sequence alignment algorithms including Optimal, and Heuristic based are Needleman and Wunsch (Needleman and Wunsh, 1970), GLASS (Delcher, *et al*, 1999), WABA (Kent and Zahler, 2000), AVID (Bray, *et al*, 2002), AlignMe (Stamm, *et al*, 2013) and GLASS (Brudno and Morgenstern, 2002), (Batzoglou, *et al*, 2000), (Young, 1989). We will not be discussing global alignment algorithms in this paper.

The alignment algorithms performances in the literature are based off several key measurements. This is very touchy, subjective topic and can vary from algorithm to algorithm. Some are based on type of sequence (they can be only for DNA, or RNA), length (some algorithms can be good for shorter than others), Measure of accuracy (since there is no standard here, this is a controversial and subjective), Speed of alignment (time taken to align the sequences in study) and lastly memory efficiency (how much memory is taken to find the alignment) (Reddy and Fields, 2020).

Pairwise sequence alignment algorithm forms the basis for multiple sequence alignment (Reddy and Fields, 2022) and lately AI (Reddy and Fields, 2022) has been used on these algorithms and set of algorithms are emerging in this field. Multiple sequence alignment algorithms are beyond the scope of this paper.

## 2. Background

In this section we will talk about the popular algorithms in local sequence alignment. Our algorithm is greatly influenced by previous algorithms and has taken clues and has questioned at times on the approach and ideas of the previous algorithms. We first start the section with optimal and then heuristic algorithms.

Smith-Waterman Algorithm is an optimal local sequence alignment algorithm, employing a technique called '*Dynamic Programming',* where a problem is broken into smaller problems and solving these smaller problems recursively (Reddy and Fields, 2020). The solutions to the subproblems are saved and are brought together to find the solution to the entire problem.

This optimal algorithm produces an optimal local sequence alignment between two sequences S1 and S2 of length m and n, respectively, in time and space equal to O(mn) (Reddy and Fields, 2020). As the sequence length increase the performance decreases exponentially.

To overcome short comings of optimal algorithm, Heuristic algorithms were developed later. Heuristic algorithms find a near-optimal solutions sacrificing little sensitivity for speed. Since it is near perfect and can calculate long sequences running in billions of characters, heuristic algorithms are preferred. All heuristic algorithms run in stages, where they find maximum subsequence and try to find regions arounds them to get sequences local sequence alignment. These subsequences are called seeds or anchors depending on whether the subsequence is one large word (anchor) or small words made of few characters (seed).

The first heuristic algorithm we are going to talk is FASTA, which stands for FAST-ALL is a heuristic algorithm developed by Lipman and Pearson (Lipman and Peason, 1985), (Reddy and Fields, 2020). FASTA uses a look-up table to find perfect subsequence matches of size 'l' and then hashing them. Time is saved for searching seeds of length 'l', then proceeds to find such seeds in a diagonal path, since the final alignment is more likely to be found in this diagonal length. It then uses a directed weighted graph for regions in between the seeds along the best diagonal found to stitch the seeds. The advantage of FASTA over Optimal algorithm is the speed but if there are 2 optimal diagonals or if the seeds are smaller than the size deployed in FASTA, then the algorithm loses considerable sensitivity (Reddy and Fields, 2020).

Basic Local Alignment Search Tool BLAST (Altschul, *et al*, 1990), uses a look-up table to identify seeds and is faster than FASTA (Reddy and Fields, 2020). Sliding window technique is employed to find all good neighbors seeds for each seed it finds in both directions (Reddy and Fields, 2020). When all seeds are found, it then proceeds to find the seeds (HSP, high scoring pairs), and extend them until they fall under a threshold score 'k'. These HSP are then stitched using a restricted dynamic programming which is a version of Smith-Waterman Algorithm (Reddy and Fields, 2020).

BLAT – BLAST like alignment tool (Kent, 2002) is much faster than BLAST. BLAT differs from previous algorithms in the way sequences are indexed. "non-overlapping seeds of S2 are run through the database of sequence and then a new scan is run linearly through the S1, whereas BLAST builds an index of S1 and then scans linearly through the database" (Kent, 2002). This saves time. After this stage, it then searches for seeds with some mismatch's 'n' in them around the seeds it found earlier. Then the HSPs of seeds and mismatch seeds are extended like BLAST to form a final alignment. As with BLAST, BLAT cannot find smaller homologous regions as the seeds taken as not small enough.

BLASTZ (Schwartz, *et al*, 2000), is the fastest among the BLAST family of algorithms, employs a different method. All repeats in the sequence are removed (Reddy and Fields, 2020). It then looks for seeds of length 'l' with

almost one-character transition. All seeds are then extended on both sides. For regions in between the seeds, it employs smaller seeds and uses optimal alignment to stitch these seeds to form final alignment. Since repeat seeds are not used again, and transition seeds set to almost one, there is a possibility that this algorithm performs poorly when it is used for divergent sequences. Meaning, say a drosophila and a pig DNA.

PatternHunter (Ma, *Et al*, 2002) introduced a seed called spaced seed to further improve the sensitivity and speed. It uses a combination of priority queues variation of red-black tree, queue, and hash table to achieve speed (Ma, *Et al*, 2002). A spaced seed is a generic seed which is converted from A, C, G, T to Binary form e.g.: 1010101,4 where 1 is a match and 4 is a score (Reddy and Fields, 2020). It then finds the best diagonal as in FASTA to find the final alignment (Reddy and Fields, 2020). The algorithm is written in JAVA, and encounters memory problems for long sequences.

UBlast (Edgar, 2010) introduces a new technique by finding fewer good hits. Meaning, subsequences which are found least but are long, to improve speed on BLAST and MEGABLAST (Zhang Z, *et al*, 2000) which is algorithm from BLAST family. The technique is targeting more speed than sensitivity.

LAST (Kiełbasa, *et al*, 2011) is recent algorithm. It uses adaptive alignment seeds; these adaptive seeds vary in length and the number of indels in them. So adaptive seeds can be of different lengths and weight. By weight, a score associated with the seed. The rest of the algorithm is very similar to BLAST. AllAlign (Wachtel, 2016) is a new algorithm developed, however literature of this AWS based web algorithm is very limited.

LAMBDA (Singer, *et al*, 2014) is new algorithm for protein alignment. It implements a technique where there are more than 1 protein sequences as the target sequences to be aligned with a pre indexed database set of all other know sequences (Reddy and Fields, 2020). It is optimized for big or large biological data and uses a Suffix tree to get the maximal common subsequences or maximal unique sequences as our algorithm in this paper, amongst them and then goes about aligning these subsequences against a pre indexed database (pre indexed based off suffix array) (Reddy and Fields, 2020).

MASAA ((Reddy, 2009) [3] introduced in 2008 is based on Ukkonen suffix (Reddy and Fields, 2020) tree. The algorithm uses double indexing and back tracking and identifies maximum match subsequences (MMSS) (Reddy and Fields, 2020). In the later stages, it stitches the MMSS regions to get the final alignment.

MASAA – S (Reddy and Fields, 2020) introduced in 2019 which also uses suffix trees, uses a threshold seed in between the MUMs . The algorithm we are proposing is more sensitive than MASAA. In our recent algorithm which was accepted in the Journal of Biosciences and Engineering, (BIOEJ), we used adaptive seeds, however in this algorithm we want to use both adaptive seeds and perfect seeds to improve sensitivity and see then how far we can further the sensitivity without sacrificing speed.

## 3. Motivation

The motivation for this paper is our previous paper which was accepted in the Journal of Biosciences and Technology, we used similar technique of first finding maximal matches, as it is easy to find using suffix trees and forms base of algorithm, and this step is like other algorithms in literature. This paper addresses the pivot or furthers the findings, as to how far can we go in size of adaptive seeds and perfect match seeds in between the maximal match without greatly sacrificing the speed. We think we are going to hit threshold when it comes to sensitivity as smaller seeds can increase sensitivity at a great cost of speed. In this paper, we introduce an algorithm technique which extends our algorithm (Suchindra and Nagaraj, 2023) and we introduce adaptive seeds and perfect seed match in between the MUMs to make it more sensitive while trying to keep the relatively speed of our existing algorithm. We call our algorithm Adaptive Seed and Maximal Match Local Sequence Alignment Algorithm (ASMMS).

## 4. ASMMS (Adaptive Seed and Maximal Match Local Sequence Alignment Algorithm)

ASMMS algorithm has five stages, but it completely differs in the kind of seeds selected in between the maximum match subsequences (MMSS), extension anchor stage and final stage - stitching the whole alignment from the algorithms in the literature. We will explain in detail in the coming sections.

### *4.1 Finding MMSSs*

In this first stage, the given two sequence are merged into one big sequence by introducing a special character in between the two sequences. This is done to find the Maximal Matches between the two sequences using the suffix tree. All maximal matches whose length '*l*' is greater than the threshold is selected. Initially, an arbitrary length is set at 1/3rd the length of the longest MMSS. It then gradually increases the threshold up until the length '*l*' is greater than half the length of the longest.

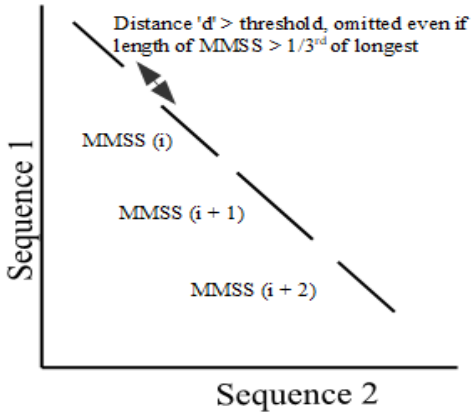


**Fig 2:** MMSS in a good neighborhood are considered (Suchindra and Nagaraj, 2023)

Based on the number of Maximal matches it gets while varying this threshold, it picks the length '*l*', when the maximum number does not change while varying from 1/3rd to say 40% of the longest MMSS anchors within a 60% of the length. All non-overlapping and non-crossing are chosen, in this step we want to select all anchors which fall in our neighborhood.

*4.2 Finding adaptive seeds In Between MMSSs*

This step can be broken down into 2 steps.

1) In this step, we find all the adaptive seeds which are found using the Suffix tree. These adaptive seeds are set at a size 20 with 4 mismatches. This step also finds more seeds while keeping the sensitivity. This also aids in speed.
2) In this step, we find perfect match seeds of length 12 in between the adaptive seeds and keep reducing the seed length up to 8 in length. These seeds should be within 1/3rd the distance from the adaptive seeds found. The size of seed at which it terminates is fixed in the algorithm primarily because, a K-mer of size between 8 or less is more likely to find more seeds than a K-mer of size ranging from 12 - 48 (Zhang Z, *et al*, 2000) (Kiełbasa , *et al*, 2013) where the authors have clearly established that there is a great propensity to get more hits when the size is between 8 – 12 (Suchindra and Nagaraj, 2023). Both the steps are shown in Fig 3.

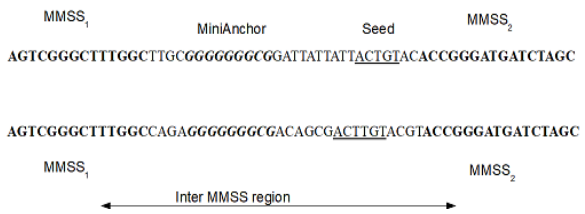


**Fig 3:** Inter MMSS, Mini Anchor and small seed (Suchindra and Nagaraj, 2023)

*4.3 Finding non criss crossing adaptive and perfect seeds*

This algorithm finds all overlapping and non-crossing matching from the adaptive seeds and perfect seed selection in the previous step. To identify anchors from overlapping and crossing matches, we use heuristic of "closeness". The question of overlapping does not arise in case of adaptive seeds because all overlapping seeds would amount to a bigger seed which would be picked up already using the suffix tree in the previous step part 1. We find that there is chance for the heuristic of closeness to arise only in the case of selecting perfect seeds of length 8 and 10 either in between maximal matches (MMSS) and Mini-Anchors or in between MMSSs. Hence this is relatively time-consuming part when compared to other previous algorithms.

*4.4 Final Stitching*

The final stitching stitches all maximal matches, adaptive seeds, and perfect seeds to make one long lengthy local sequence alignment. The maximal matches, adaptive and perfect seeds are all joined in this stage (Reddy and Fields, 2020) (Suchindra and Nagaraj, 2023). After completion of this stage, algorithm terminates, and the result is produced.

## 5. Experimental Results

Our experiments consisted of randomly generated sequences of length from, 100k to 500k and compared the speed of alignment. For smaller sequences the speed is much faster than BLASTZ. However, when the sequences grow, then the speed of our new algorithm gets closer to BLASTZ and might be slower, we attribute this behavior to finding perfect seeds in between the maximal matches, which is an iterative process.

We did not compare the BLASTZ, MASAA and ASMMS with LAMBDA, AllAlign for reason, because it needed an index database first and for a randomly generated sequences it was challenging (Reddy and Fields, 2020) and we could not download MASAA. For AllAlign, we could not find source code to download and compare, checking the performance of the sequence on a server was not clinical. Although we know LAMBDA is 500x faster than BLASTZ, here we assume that LAMBDA is faster than ours too.

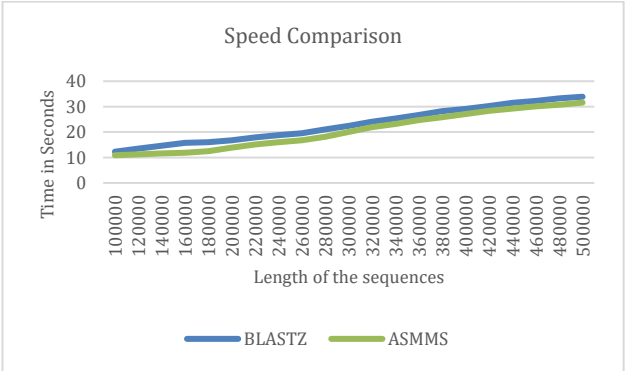


**Fig 5:** Speed Comparison on different sequence length

For sensitivity, we compared exon coverage from all three algorithms. We compared performance of these algorithms on gene dataset. Table 1 shows percentage of exon coverage, here we think ASMMS performs better than MASAA and BLASTZ, we attribute this to smaller anchors seeds being chosen once the Large Anchors are already chosen in the first step. Also, smaller seeds of different length <=12 plays a role. This step is little performance heavy (speed) as we see in the Figure 4.

**Table 1** Sensitivity comparison on genes

| Algorithm | % of exon coverage | | |
|---|---|---|---|
| | ***100 exon*** | ***90 exon*** | ***70 exon*** |
| BLASTZ | 94 | 97 | 98 |
| MASAA | 94 | 94 | 96 |
| ASMMS | 94 | 98 | 99 |

ASMMS performed better than BLASTZ and MASAA on divergent sequences, these sequences are small genes of

a drosophila fruit fly, and we compared the percentage of exon coverage. ASMMS performed better than MASAA and BLASTZ. We attribute this performance to perfect match seeds of smaller size in between MM's. This is shown in Figure 6.

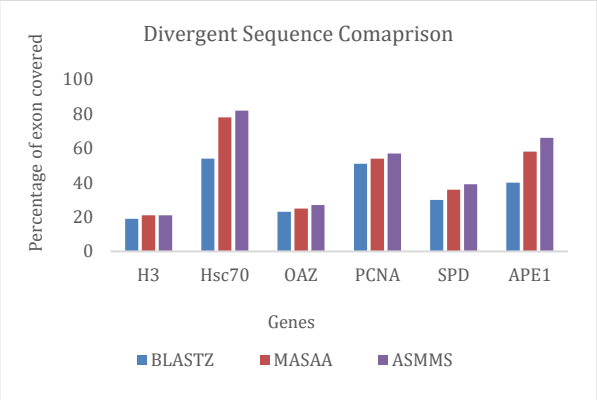


**Fig 6:** Exon coverage

## Conclusions

Pairwise sequence algorithms are very important and therefore has been an active field of research. It is also due to many private firms employing computational biology for commercial purposes. This and technological revolution in computer hardware are opened the gates where in previous older algorithms which were sensitive but slower now can be made faster using superior hardware. Although most of the algorithms concentrate in seed finding techniques, database preprocessing or hardware improvement, we believe that we have not seen the end of any of the above 3 strategies yet. Therefore, we believe there is enough research to find new data structure to speed up the seed's identification and new seeds but also enough research in parallelizing above algorithms for better performance employing clusters in the future.

## References

[1]. Reddy, Bharath Govinda. Multiple Anchor Staged Local Sequence Alignment Algorithm-MASAA. University of Northern British Columbia, 2009.

[2]. Haque, Waqar, Aravind Alex and Reddy, Bharath. 2009. Pairwise sequence alignment algorithms: a survey. In Proceedings of the 2009 conference on Information Science, Technology and Applications (ISTA '09). ACM, New York, NY, USA, 96-103. DOI: http://dx.doi.org/10.1145/1551950. 1551980

[3]. Haque, W., Aravind, A.A., & Reddy, B. (2008). An efficient algorithm for local sequence alignment. 2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 1367-1372.

[4]. Smith, T.F. and Waterman, M.S. 1981. Identification of common molecular subsequences. J. Mol. Biol. 147:195-197.

[5]. D. Lipman and W. Pearson. Rapid and sensitive protein similarity searches. Science, 227: 1435, 1441, 1985.

[6]. Altschul, S.F.,Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. 1990. Basic local alignment search tool. J. Mol. Biol. 215: 403-410.

[7]. S. F. Altschul, T. L. Madden, A. A. Schaer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Research,25(17): 3389, 3392, 1997.

[8]. Schwartz, S., Zhang, Z., Frazer, K.A., Smit, A., Riemer, C., Bouck, J., Gibbs, R., Hardison, R., and Miller, W. 2000. PipMaker-A web server for aligning two genomic DNA sequences. Genome Res. 10: 577-586.

[9]. Ma, B., J. Tromp, and M. Li (2002). Patternhunter: Faster and more sensitive homology search. Bioinformatics 18, 440-445.

[10]. No, Laurent and Kucherov, Gregory. YASS: enhancing the sensitivity of DNA similarity search. Nucleic Acids Res, 2005.

[11]. Singer. H. Hauswedell, , J. & Reinert, K. Lambda: the local aligner for massive biological data. Bioinformatics 30, i349–i355 (2014).

[12]. Edgar (2010) Edgar RC. Search and clustering orders of magnitude faster than BLAST. Bioinformatics. 2010;26(19):2460–2461. doi: 10.1093/bioinformatics/btq461.

[13]. Kiełbasa SM, Wan R, Sato K, Horton P, Frith MC. Genome Res. Adaptive seeds tame genomic sequence comparison. 2011 21(3):487-93.

[14]. Stamm M, Staritzbichler R, Khafizov K, Forrest LR (2013) Alignment of Helical Membrane Protein Sequences Using AlignMe. PLOS ONE 8(3): e57731. https://doi.org/10.1371/journal.pone.0057731

[15]. Wachtel. E. All ALL Local Alignment. http://www.allalign.com/

[16]. Zhang Z, *et al*. Agreedy algorithm for aligning DNAsequences, J. Comp. Biol , 2000, vol. 7 (pg. 203-214)

[17]. Kent. W. James, BLAT: The BLAST-Like Alignment Tool. Genome Research, Vol. 12, Issue 4, 656-664, 2002.

[18]. Needleman, S.B. and Wunsch, C.D. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. J. Mol. Biol. 48: 443-453.

[19]. Delcher A.L., Kasif S., Fleischmann R.D., Peterson J., White O., and Salzberg S.L. 1999. Alignment of whole genomes. Nucleic Acids Res. 27: 2369-2376.

[20]. Kent, J. and Zahler, M. 2000. The Intronerator: Exploring introns and alternative splicing in C. elegans genomic alignment. Genome Res. 10: 1115-1125.

[21]. Bray Nick, Dubchak Inna and Pachter Lior, Avid: A global alignment program, Genome Research. 2003 13: 97-102; 2002.

[22]. Brudno, M. and Morgenstern, B. Fast and sensitive alignment of large genomic sequences, 2002.

[23]. Batzoglou, S., Pachter, L., Mesirov, J.P., Berger, B., and Lander, E.S. 2000. Human and mouse gene structure: Comparative analysis and application to exon prediction. Genome Res. 10: 950-958.

[24]. M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[25]. Reddy, B, Fields, R. 2020, Multiple Anchor Staged alignment algorithm – Sensitive, 2020, In proceedings with The International Conference on Information and Computer Technologies (ICICT), 2020, San Jose. USA.

[26]. Suchindra, Suchindra & Nagaraj, P (2023). New sequence alignment algorithm using AI rules and dynamic seeds. Accepted in the proceeding of International Journal of Current Engineering and Technology, 2023.

[27]. Reddy, B., & Fields, R. (2022). Multiple sequence alignment algorithms in bioinformatics. In Smart Trends in Computing and Communications: Proceedings of SmartCom 2021 (pp. 89-98). Springer Singapore.

[28]. Reddy, B., & Fields, R. (2022, April). From past to present: a comprehensive technical review of rule-based expert systems from 1980--2021. In Proceedings of the 2022 ACM Southeast Conference (pp. 167-172).